CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition

Xuejing Yuan[1,2], Yuxuan Chen[3], Yue Zhao[1,2], Yunhui Long[4], Xiaokang Liu[1,2], Kai Chen*[1,2], Shengzhi Zhang[3,5], Heqing Huang , XiaoFeng Wang[6], and Carl A. Gunter[4]

[1]SKLOIS, Institute of Information Engineering, Chinese Academy of Sciences, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, China
[3]Department of Computer Science, Florida Institute of Technology, USA
[4]Department of Computer Science, University of Illinois at Urbana-Champaign, USA
[5]Department of Computer Science, Metropolitan College, Boston University, USA
[6]School of Informatics and Computing, Indiana University Bloomington, USA

Abstract

The popularity of automatic speech recognition (ASR) systems, like Google Assistant, Cortana, brings in security concerns, as demonstrated by recent attacks. The impacts of such threats, however, are less clear, since they are either less stealthy (producing noise-like voice commands) or requiring the physical presence of an attack device (using ultrasound speakers or transducers). In this paper, we demonstrate that not only are more practical and surreptitious attacks feasible but they can even be *automatically* constructed. Specifically, we find that the voice commands can be stealthily embedded into songs, which, when played, can effectively control the target system through ASR without being noticed. For this purpose, we developed novel techniques that address a key technical challenge: integrating the commands into a song in a way that can be effectively recognized by ASR through the air, in the presence of background noise, while not being detected by a human listener. Our research shows that this can be done automatically against real world ASR applications[1]. We also demonstrate that such *CommanderSongs* can be spread through Internet (e.g., YouTube) and radio, potentially affecting millions of ASR users. Finally we present mitigation techniques that defend existing ASR systems against such threat.

1 Introduction

Intelligent voice control (IVC) has been widely used in human-computer interaction, such as Amazon Alexa [1], Google Assistant [6], Apple Siri [3], Microsoft Cortana [14] and iFLYTEK [11]. Running the state-of-the-art ASR techniques, these systems can effectively interpret natural voice commands and execute the corresponding operations such as unlocking the doors of home or cars, making online purchase, sending messages, and etc. This has been made possible by recent progress in machine learning, deep learning [31] in particular, which vastly improves the accuracy of speech recognition. In the meantime, these deep learning techniques are known to be vulnerable to adversarial perturbations [37, 21, 27, 25, 20, 49, 28, 44]. Hence, it becomes imperative to understand the security implications of the ASR systems in the presence of such attacks.

Threats to ASR Prior research shows that carefully-crafted perturbations, even a small amount, could cause a machine learning classifier to misbehave in an unexpected way. Although such adversarial learning has been extensively studied in image recognition, little has been done in speech recognition, potentially due to the new challenge in this domain: unlike adversarial images, which include the perturbations of less noticeable background pixels, changes to voice commands often introduce noise that a modern ASR system is designed to filter out and therefore cannot be easily misled.

Indeed, a recent attack on ASR utilizes noise-like hidden voice command [22], but the white box attack is based on a traditional speech recognition system that uses a Gaussian Mixture Model (GMM), not the DNN behind today's ASR systems. Another attack transmits inaudible commands through ultrasonic sound [53], but it exploits microphone hardware vulnerabilities instead of the weaknesses of the DNN. Moreover, an attack device, e.g., an ultrasonic transducer or speaker, needs to be placed close to the target ASR system. So far little success has been reported in generating "adversarial sound" that practically fools deep learning technique but remains inconspicuous to human ears, and meanwhile allows it to be played from the remote (e.g., through YouTube) to attack a large number of ASR systems.

To find *practical* adversarial sound, a few technical challenges need to be addressed: (C1) the adversarial audio sample is expected to be effective in a complicated, real-world audible environment, in the presence of elec-

*Corresponding author: chenkai@iie.ac.cn

[1]Demos of attacks are uploaded on the website (https://sites.google.com/view/commandersong/)

tronic noise from speaker and other noises; (C2) it should be stealthy, unnoticeable to ordinary users; (C3) impactful adversarial sound should be remotely deliverable and can be played by popular devices from online sources, which can affect a large number of IVC devices. All these challenges have been found in our research to be completely addressable, indicating that the threat of audio adversarial learning is indeed realistic.

CommanderSong. More specifically, in this paper, we report a practical and systematic adversarial attack on real world speech recognition systems. Our attack can *automatically* embed a set of commands into a (randomly selected) song, to spread to a large amount of audience (addressing C3). This revised song, which we call *CommanderSong*, can sound completely normal to ordinary users, but will be interpreted as commands by ASR, leading to the attacks on real-world IVC devices. To build such an attack, we leverage an open source ASR system Kaldi [13], which includes acoustic model and language model. By carefully synthesizing the outputs of the acoustic model from both the song and the given voice command, we are able to generate the adversarial audio with minimum perturbations through gradient descent, so that the CommanderSong can be less noticeable to human users (addressing C2, named WTA attack). To make such adversarial samples practical, our approach has been designed to capture the electronic noise produced by different speakers, and integrate a generic noise model into the algorithm for seeking adversarial samples (addressing C1, called WAA attack).

In our experiment, we generated over 200 CommanderSongs that contain different commands, and attacked Kaldi with an 100% success rate in a WTA attack and a 96% success rate in a WAA attack. Our evaluation further demonstrates that such a CommanderSong can be used to perform a *black box* attack on a mainstream ASR system iFLYTEK[2] [11] (neither source code nor model is available). iFLYTEK has been used as the voice input method by many popular commercial apps, including WeChat (a social app with 963 million users), Sina Weibo (another social app with 530 million users), JD (an online shopping app with 270 million users), etc. To demonstrate the impact of our attack, we show that CommanderSong can be spread through YouTube, which might impact millions of users. To understand the human perception of the attack, we conducted a user study[3] on Amazon Mechanical Turk [2]. Among over 200 participants, none of them identified the commands inside our CommanderSongs. We further developed the defense solutions against this attack and demonstrated their effectiveness.

[2]We have reported this to iFLYTEK, and are waiting for their responses.

[3]The study is approved by the IRB.

Contributions. The contributions of this paper are summarized as follows:

• *Practical adversarial attack against ASR systems*. We designed and implemented the first practical adversarial attacks against ASR systems. Our attack is demonstrated to be *robust*, working across air in the presence of environmental interferences, *transferable*, effective on a black box commercial ASR system (i.e., iFLYTEK) and *remotely deliverable*, potentially impacting millions of users.

• *Defense against CommanderSong.* We design two approaches (audio turbulence and audio squeezing) to defend against the attack, which proves to be effective by our preliminary experiments.

Roadmap. The rest of the paper is organized as follows: Section 2 gives the background information of our study. Section 3 provides motivation and overviews our approach. In Section 4, we elaborate the design and implementation of CommanderSong. In Section 5, we present the experimental results, with emphasis on the difference between machine and human comprehension. Section 6 investigates deeper understanding on CommanderSongs. Section 7 shows the defense of the CommanderSong attack. Section 8 compares our work with prior studies and Section 9 concludes the paper.

2 Background

In this section, we overview existing speech recognition system, and discuss the recent advance on the attacks against both image and speech recognition systems.

2.1 Speech Recognition

Automatic speech recognition is a technique that allows machines to recognize/understand the semantics of human voice. Besides the commercial products like Amazon Alexa, Google Assistant, Apple Siri, iFLYTEK, etc., there are also open-source platforms such as Kaldi toolkit [13], Carnegie Mellon University's Sphinx toolkit [5], HTK toolkit [9], etc. Figure 1 presents an overview of a typical speech recognition system, with two major components: feature extraction and decoding based on pre-trained models (e.g., acoustic models and language models).

After the raw audio is amplified and filtered, acoustic features need to be extracted from the preprocessed audio signal. The features contained in the signal change significantly over time, so short-time analysis is used to evaluate them periodically. Common acoustic feature extraction algorithms include Mel-Frequency Cepstral Coefficients (MFCC) [40], Linear Predictive Coefficient (LPC) [34], Perceptual Linear Predictive (PLP) [30], etc. Among them, MFCC is the most frequently used one in

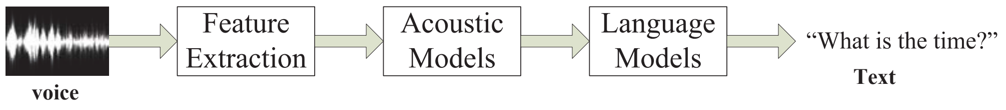


Figure 1: Architecture of Automatic Speech Recognition System.

both open source toolkit and commercial products [42]. GMM can be used to analyze the property of the acoustic features. The extracted acoustic features are matched against pre-trained acoustic models to obtain the likelihood probability of phonemes. Hidden Markov Models (HMM) are commonly used for statistical speech recognition. As GMM is limited to describe a non-linear manifold of the data, Deep Neural Network-Hidden Markov Model (DNN-HMM) has been widely used for speech recognition in academic and industry community since 2012 [32].

Recently, end-to-end deep learning becomes used in speech recognition systems. It applies a large scale dataset and uses CTC (Connectionist Temporal Classification) loss function to directly obtain the characters rather than phoneme sequence. CTC locates the alignment of text transcripts with input speech using an all-neural, sequence-to-sequence neural network. Traditional speech recognition systems involve many engineered processing stages, while CTC can supersede these processing stages via deep learning [17]. The architecture of end-to-end ASR systems always includes an encoder network corresponding to the acoustic model and a decoder network corresponding to the language model [47]. DeepSpeech [17] and Wav2Letter [24] are popular open source end-to-end speech recognition systems.

2.2 Existing Attacks against Image and Speech Recognition Systems

Nowadays people are enjoying the convenience of integrating image and speech as new input methods into mobile devices. Hence, the accuracy and dependability of image and speech recognition pose critical impact on the security of such devices. Intuitively, the adversaries can compromise the integrity of the training data if they have either physical or remote access to it. By either revising existing data or inserting extra data in the training dataset, the adversaries can certainly tamper the dependability of the trained models [38].

When adversaries do not have access to the training data, attacks are still possible. Recent research has been done to deceive image recognition systems into making wrong decision by slightly revising the input data. The fundamental idea is to revise an image slightly to make it "look" different from the views of human being and machines. Depending on whether the adversary knows the algorithms and parameters used in the recognition systems, there exist white box and black box attacks. Note that the adversary always needs to be able to interact with the target system to observe corresponding output for any input, in both white and black box attacks. Early researches [50, 48, 19] focus on the revision and generation of the digital image file, which is directly fed into the image recognition systems. The state-of-the-art researches [37, 21, 27] advance in terms of practicality by printing the adversarial image and presenting it to a device with image recognition functionality.

However, the success of the attack against image recognition systems has not been ported to the speech recognition systems until very recently, due to the complexity of the latter. The speech, a time-domain continuous signal, contains much more features compared to the static images. Hidden voice command [22] launched both black box (i.e., inverse MFCC) and white box (i.e., gradient decent) attacks against speech recognition systems, and generated obfuscated commands to ASR systems. Though seminal in attacking speech recognition systems, it is also limited to make practical attacks. For instance, a large amount of human effort is involved as feedback for the black box approach, and the white box approach is based on GMM-based acoustic models, which have been replaced by DNN-based ones in most modern speech recognition systems. The recent work DolphinAttack [53] proposed a completely inaudible voice attack by modulating commands on ultrasound carriers and leveraging microphone vulnerabilities (i.e., the nonlinearity of the microphones). As noted by the authors, such attack can be eliminated by an enhanced microphone that can suppress acoustic signals on ultrasound carrier, like iPhone 6 Plus.

3 Overview

In this section, we present the motivation of our work, and overview the proposed approach to generate the practical adversarial attack.

3.1 Motivation

Recently, adversarial attacks on image classification have been extensively studied [21, 27]. Results show that even the state-of-the-art DNN-based classifier can be fooled by small perturbations added to the original image [37], producing erroneous classification results. However, the impact of adversarial attacks on the most advanced speech recognition systems, such as those integrating DNN models, has never been systematically studied. Hence, in this paper, we investigated DNN-based speech recognition systems, and explored adversarial attacks against them. Researches show that commands can be transmitted to IVC devices through inaudible ultrasonic sound [53] and noises [22]. Even though the existing works against ASR systems are seminal, they are limited in some aspects. Specifically, ultrasonic sound can be defeated by using a low-pass filter (LPF) or analyzing the signal frequency range, and noises are easy to be noticed by users.

Therefore, the research in this paper is motivated by the following questions: (Q1) Is it possible to build the practical adversarial attack against ASR systems, given the facts that the most ASR systems are becoming more intelligent (e.g., by integrating DNN models) and that the generated adversarial samples should work in the very complicated physical environment, e.g., electronic noise from speaker, background noise, etc.? (Q2) Is it feasible to generate the adversarial samples (including the target commands) that are difficult, or even impossible, to be noticed by ordinary users, so the control over the ASR systems can happen in a "hidden" fashion? (Q3) If such adversarial audio samples can be produced, is it possible to impact a large amount of victims in an automated way, rather than solely relying on attackers to play the adversarial audio and affecting victims nearby? Below, we will detail how our attack is designed to address the above questions.

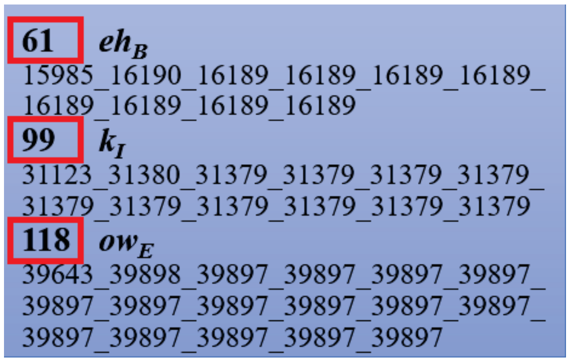
3.2 The Philosophy of Designing Our Attack

To address Q3, our idea is to choose songs as the "carrier" of the voice commands recognizable by ASR systems. The reason of choosing such "carrier" is at least two-fold. On one hand, enjoying songs is always a preferred way for people to relax, e.g., listening to the music station, streaming music from online libraries, or just browsing YouTube for favorite programs. Moreover, such entertainment is not restricted by using radio, CD player, or desktop computer any more. A mobile device, e.g., Android phone or Apple iPhone, allows people to enjoy songs everywhere. Hence, choosing the song as the "carrier" of the voice command automatically helps impact millions of people. On the other hand, "hiding" the desired command in the song also makes the command much more difficult to be noticed by victims, as long as Q2 can be reasonably addressed. Note that we do not rely on the lyrics in the song to help integrate the desired command. Instead, we intend to avoid the songs with the lyrics similar to our desired command. For instance, if the desired command is "open the door", choosing a song with the lyrics of "open the door" will easily catch the victims' attention. Hence, we decide to use random songs as the "carrier" regardless of the desired commands.

Actually choosing the songs as the "carrier" of desired commands makes Q2 even more challenging. Our basic idea is when generating the adversarial samples, we revise the original song leveraging the pure voice audio of the desired command as a reference. In particular, we find the revision of the original song to generate the adversarial samples is always a trade off between preserving the fidelity of the original song and recognizing the desired commands from the generated sample by ASR systems. To better obfuscate the desired commands in the song, in this paper we emphasize the former than the latter. In other words, we designed our revision algorithm to maximally preserve the fidelity of the original song, at the expense of losing a bit success rate of recognition of the desired commands. However, such expense can be compensated by integrating the same desired command multiple times into one song (the command of "open the door" may only last for 2 seconds.), and the successful recognition of one suffices to impact the victims.

Technically, in order to address Q2, we need to investigate the details of an ASR system. As shown in Figure 1, an ASR system is usually composed of two pre-trained models: an acoustic model describing the relationship between audio signals and phonetic units, and a language model representing statistical distributions over sequences of words. In particular, given a piece of pure voice audio of the desired command and a "carrier" song, we can feed them into an ASR system separately, and intercept the intermediate results. By investigating the output from the acoustic model when processing the audio of the desired command, and the details of the language model, we can conclude the "information" in the output that is necessary for the language model to produce the correct text of the desired command. When we design our approach, we want to ensure such "information" is only a small subset (hopefully the minimum subset) of the output from the acoustic model. Then, we carefully craft the output from the acoustic model when processing the original song, to make it "include" such "information" as well. Finally, we inverse the acoustic model and the feature extraction together, to directly produce the adversarial sample based on the crafted output (with the "information" necessary for the language model to produce the correct text of the desired command).

Theoretically, the adversarial samples generated above can be recognized by the ASR systems as the desired command if directly fed as input to such systems. Since such input usually is in the form of a wave file (in "WAV" format) and the ASR systems need to expose APIs to accept the input, we define such attack as the WAV-To-API (WTA) attack. However, to implement a practical



```
61  eh_B
15985_16190_16189_16189_16189_16189_
16189_16189_16189_16189
99  k_I
31123_31380_31379_31379_31379_31379_
31379_31379_31379_31379_31379_31379
118  ow_E
39643_39898_39897_39897_39897_39897_
39897_39897_39897_39897_39897_39897_
39897_39897_39897_39897_39897
```

Figure 2: Result of decoding "Echo".

attack as in Q1, the adversarial sample should be played by a speaker to interact with IVC devices over the air. In this paper, we define such practical attack as WAV-Air-API (WAA) attack. The challenge of the WAA attack is when playing the adversarial samples by a speaker, the electronic noise produced by the loudspeakers and the background noise in the open air have significant impact on the recognition of the desired commands from the adversarial samples. To address this challenge, we improve our approach by integrating a generic noise model to the above algorithm with the details in Section 4.3.

4 Attack Approach

We implement our attack by addressing two technical challenges: (1) minimizing the perturbations to the song, so the distortion between the original song and the generated adversarial sample can be as unnoticeable as possible, and (2) making the attack practical, which means CommanderSong should be played over the air to compromise IVC devices. To address the first challenge, we proposed *pdf-id sequence matching* to incur minimum revision at the output of the acoustic model, and use gradient descent to generate the corresponding adversarial samples as in Section 4.2. The second challenge is addressed by introducing a generic noise model to simulate both the electronic noise and background noise as in Section 4.3. Below we elaborate the details.

4.1 Kaldi Platform

We choose the open source speech recognition toolkit Kaldi [13], due to its popularity in research community. Its source code on github obtains 3,748 stars and 1,822 forks [4]. Furthermore, the corpus trained by Kaldi on "Fisher" is also used by IBM [18] and Microsoft [52].

In order to use Kaldi to decode audio, we need a trained model to begin with. There are some models on Kaldi website that can be used for research. We took advantage of the "ASpIRE Chain Model" (referred as "ASpIRE model" in short), which was one of the latest released decoding models when we began our study[4]. After manually analyzing the source code of Kaldi (about 301,636 lines of shell scripts and 238,107 C++ SLOC), we completely explored how Kaldi processes audio and decodes it to texts. Firstly, it extracts acoustic features like MFCC or PLP from the raw audio. Then based on the trained probability density function (p.d.f.) of the acoustic model, those features are taken as input to DNN to compute the posterior probability matrix. The p.d.f. is indexed by the pdf identifier (pdf-id), which exactly indicates the column of the output matrix of DNN.

Phoneme is the smallest unit composing a word. There are three states (each is denoted as an HMM state) of sound production for each phoneme, and a series of transitions among those states can identify a phoneme. A transition identifier (transition-id) is used to uniquely identify the HMM state transition. Therefore, a sequence of transition-ids can identify a phoneme, so we name such a sequence as *phoneme identifier* in this paper. Note that the transition-id is also mapped to pdf-id. Actually, during the procedure of Kaldi decoding, the phoneme identifiers can be obtained. By referring to the pre-obtained mapping between transition-id and pdf-id, any phoneme identifier can also be expressed as a specific sequence of pdf-ids. Such a specific sequence of pdf-ids actually is a segment from the posterior probability matrix computed from DNN. This implies that to make Kaldi decode any specific phoneme, we need to have DNN compute a posterior probability matrix containing the corresponding sequence of pdf-ids.

Table 1: Relationship between transition-id and pdf-id.

Phoneme	HMM-state	Pdf-id	Transition-id	Transition
eh_B	0	6383	15985	0→1
			15986	0→2
eh_B	1	5760	16189	self-loop
			16190	1→2
k_I	0	6673	31223	0→1
			31224	0→2
k_I	1	3787	31379	self-loop
			31380	1→2
ow_E	0	5316	39643	0→1
			9644	0→2
ow_E	1	8335	39897	self-loop
			39898	1→2

[4]There are three decoding models on Kaldi platform currently. ASpIRE Chain Model we used in this paper was released on October 15th, 2016, while SRE16 Xvector Model was released on October 4th, 2017, which was not available when we began our study. The CVTE Mandarin Model, released on June 21st 2017 was trained in Chinese [13].

To illustrate the above findings, we use Kaldi to process a piece of audio with several known words, and obtain the intermediate results, including the posterior probability matrix computed by DNN, the transition-ids sequence, the phonemes, and the decoded words. Figure 2 demonstrates the decoded result of *Echo*, which contains three phonemes. The red boxes highlight the id representing the corresponding phoneme, and each phoneme is identified by a sequence of transition-ids, or the *phoneme identifiers*. Table 1 is a segment from the the relationship among the phoneme, pdf-id, transition-id, etc. By referring to Table 1, we can obtain the pdf-ids sequence corresponding to the decoded transition-ids sequence[5]. Hence, for any posterior probability matrix demonstrating such a pdf-ids sequence should be decoded by Kaldi as eh_B.

4.2 Gradient Descent to Craft Audio

Figure 3 demonstrates the details of our attack approach. Given the original song $x(t)$ and the pure voice audio of the desired command $y(t)$, we use Kaldi to decode them separately. By analyzing the decoding procedures, we can get the output of DNN matrix $\mathbf{A}$ of the original song (Step ① in Figure 3) and the phoneme identifiers of the desired command audio (Step ④ in Figure 3).

The DNN's output A is a matrix containing the probability of each pdf-id at each frame. Suppose there are n frames and k pdf-ids, let $a_{i,j}$ $(1 \le i \le n, 1 \le j \le k)$ be the element at the ith row and jth column in $\mathbf{A}$. Then $a_{i,j}$ represents the probability of the jth pdf-id at frame i. For each frame, we calculate the most likely pdf-id as the one with the highest probability in that frame. That is,

$$m_i = \arg\max_j a_{i,j}.$$

Let $\mathbf{m} = (m_1, m_2, \ldots, m_n)$. $\mathbf{m}$ represents a sequence of most likely pdf-ids of the original song audio $x(t)$. For simplification, we use g to represent the function that takes the original audio as input and outputs a sequence of most likely pdf-ids based on DNN's predictions. That is,

$$g(x(t)) = \mathbf{m}.$$

As shown in Step ⑤ in Figure 3, we can extract a sequence of pdf-id of the command $\mathbf{b} = (b_1, b_2, \ldots, b_n)$, where b_i $(1 \le i \le n)$ represents the highest probability pdf-id of the command at frame i. To have the original song decoded as the desired command, we need to identify the minimum modification $\delta(t)$ on $x(t)$ so that $\mathbf{m}$ is same or close to $\mathbf{b}$. Specifically, we minimize the $L1$ distance between $\mathbf{m}$ and $\mathbf{b}$. As $\mathbf{m}$ and $\mathbf{b}$ are related with the pdf-id sequence, we define this method as *pdf-id sequence matching* algorithm.

[5]For instance, the pdf-ids sequence for eh_B should be *6383, 5760, 5760, 5760, 5760, 5760, 5760, 5760, 5760, 5760.*

Based on these observations we construct the following objective function:

$$\arg\min_{\delta(t)} \quad \|g\left(x(t)+\delta(t)\right)-\mathbf{b}\|_1. \tag{1}$$

To ensure that the modified audio does not deviate too much from the original one, we optimize the objective function Eq (1) under the constraint of $|\delta(t)| \le l$.

Finally, we use gradient descent [43], an iterative optimization algorithm to find the local minimum of a function, to solve the objective function. Given an initial point, gradient descent follows the direction which reduces the value of the function most quickly. By repeating this process until the value starts to remain stable, the algorithm is able to find a local minimum value. In particular, based on our objective function, we revise the song $x(t)$ into $x'(t) = x(t) + \delta(t)$ with the aim of making most likely pdf-ids $g\left(x'(t)\right)$ equal or close to $\mathbf{b}$. Therefore, the crafted audio $x'(t)$ can be decoded as the desired command.

To further preserve the fidelity of the original song, one method is to minimize the time duration of the revision. Typically, once the pure command voice audio is generated by a text-to-speech engine, all the phonemes are determined, so as to the phoneme identifiers and $\mathbf{b}$. However, the speed of the speech also determines the number of frames and the number of transition-ids in a phoneme identifier. Intuitively, slow speech always produces repeated frames or transition-ids in a phoneme. Typically people need six or more frames to realize a phoneme, but most speech recognition systems only need three to four frames to interpret a phoneme. Hence, to introduce the minimal revision to the original song, we can analyze $\mathbf{b}$, reduce the number of repeated frames in each phoneme, and obtain a shorter $\mathbf{b}' = (b_1, b_2, \ldots, b_q)$, where $q < n$.

4.3 Practical Attack over the Air

By feeding the generated adversarial sample directly into Kaldi, the desired command can be decoded correctly. However, playing the sample through a speaker to physically attack an IVC device typically cannot work. This is mainly due to the noises introduced by the speaker and environment, as well as the distortion caused by the receiver of the IVC device. In this paper, we do not consider the invariance of background noise in different environments, e.g., grocery, restaurant, office, etc., due to the following reasons: (1) In a quite noisy environment like restaurant or grocery, even the original voice command $y(t)$ may not be correctly recognized by IVC devices; (2) Modeling any slightly variant background noise itself is still an open research problem; (3) Based on our observation, in a normal environment like home, office, lobby, the major impacts on the physical attack are the electronic noise from the speaker and the distortion from the receiver of the IVC devices, rather than the background noise.

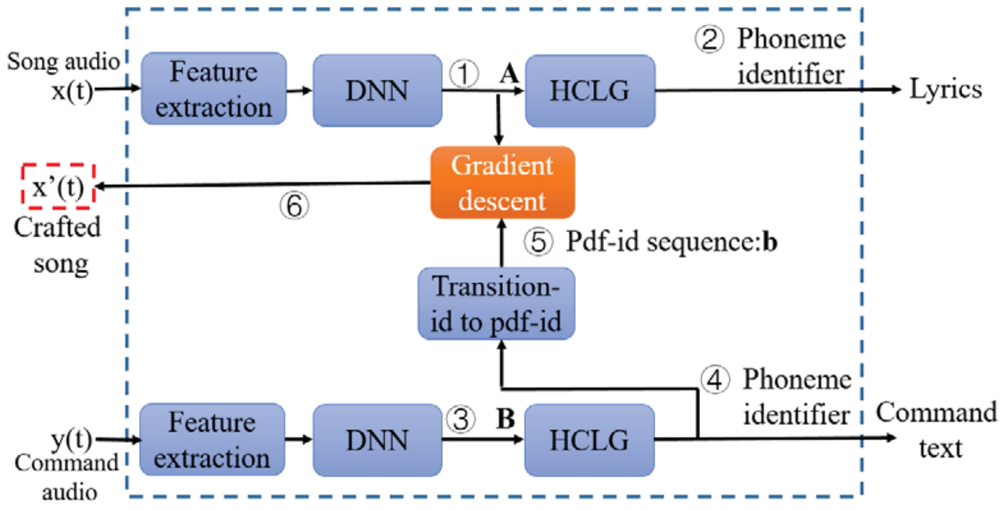


Figure 3: Steps of attack.

Hence, our idea is to build a noise model, considering the speaker noise, the receiver distortion, as well as the generic background noise, and integrate it in the approach in Section 4.2. Specifically, we carefully picked up several songs and played them through our speaker in a very quiet room. By comparing the recorded audio (captured by our receiver) with the original one, we can capture the noises. Note that playing "silent" audio does not work since the electronic noise from speakers may depend on the sound at different frequencies. Therefore, we intend to choose the songs that cover more frequencies. Regarding the comparison between two pieces of audio, we have to first manually align them and then compute the difference. We redesign the objective function as shown in Eq (2).

$$\arg\min_{\mu(t)} \quad \|g\left(x(t)+\mu(t)+n(t)\right)-\mathbf{b}\|_1, \tag{2}$$

where $\mu(t)$ is the perturbation that we add to the original song, and $n(t)$ is the noise samples that we captured. In this way, we can get the adversarial audio $x'(t) = x(t) + \mu(t)$ that can be used to launch the practical attack over the air.

Such noise model above is quite device-dependent. Since different speakers and receivers may introduce different noises/distortion when playing or receiving specific audio, $x'(t)$ may only work with the devices that we use to capture the noise. To enhance the robustness of $x'(t)$, we introduce ***random noise***, which is shown in Eq (3). Here, the function ***rand()*** returns an vector of random numbers in the interval (-N,N), which is saved as a "WAV" format file to represent $n(t)$. Our evaluation results show that this approach can make the adversarial audio $x'(t)$ robust enough for different speakers and receivers.

$$n(t) = rand(t), |n(t)| <= N. \tag{3}$$

5 Evaluation

In this section, we present the experimental results of CommanderSong. We evaluated both the WTA and WAA attacks against machine recognition. To evaluate the human comprehension, we conducted a survey examining the effects of "hiding" the desired command in the song. Then, we tested the transferability of the adversarial sample on other ASR platforms, and checked whether CommanderSong can spread through Internet and radio. Finally, we measured the efficiency in terms of the time to generate the CommanderSong. Demos of attacks are uploaded on the website (https://sites.google.com/view/commandersong/).

5.1 Experiment Setup

The pure voice audio of the desired commands can be generated by any Text-To-Speech (TTS) engine (e.g., Google text-to-speech [7], etc.) or recording human voice, as long as it can be correctly recognized by Kaldi platform. We also randomly downloaded 26 songs from the Internet. To understand the impact of using different types of songs as the carrier, we intended to choose songs from different categories, i.e., popular, rock, rap, and soft music. Regarding the commands to inject, we chose 12 commonly used ones such as "turn on GPS", "ask Capital One to make a credit card payment", etc., as shown in Table 2. Regarding the computing environment, one GPU server (1075MHz GPU with 12GB memory, and 512GB hard drive) was used.

Table 2: WTA attack results.

Command	Success rate (%)	SNR (dB)	Efficiency (frames/hours)
Okay google restart phone now.	100	18.6	229/1.3
Okay google flashlight on.	100	14.7	219/1.3
Okay google read mail.	100	15.5	217/1.5
Okay google clear notification.	100	14	260/1.2
Okay google good night.	100	15.6	193/1.3
Okay google airplane mode on.	100	16.9	219/1.1
Okay google turn on wireless hot spot.	100	14.7	280/1.6
Okay google read last sms from boss.	100	15.1	323/1.4
Echo open the front door.	100	17.2	193/1.0
Echo turn off the light.	100	17.3	347/1.5
Okay google call one one zero one one nine one two zero.	100	14.8	387/1.7
Echo ask capital one to make a credit card payment.	100	15.8	379/1.9

5.2 Effectiveness

WTA Attack. In this WTA attack, we directly feed the generated adversarial songs to Kaldi using its exposed APIs, which accept raw audio file as input. Particularly, we injected each command into each of the downloaded 26 songs using the approach proposed in Section 4.2. Totally we got more than 200 adversarial songs in the "WAV" format and sent them to Kaldi directly for recognition. If Kaldi successfully identified the command injected inside, we denote the attack as successful.

Table 2 shows the WTA attack results. Each command can be recognized by Kaldi correctly. The success rate 100% means Kaldi can decode every word in the desired command correctly. The success rate is calculated as the ratio of the number of words successfully decoded and the number of words in the desired command. Note in the case that the decoded word is only one character different than that in the desired command, we consider the word is not correctly recognized.

For each adversarial song, we further calculated the average signal-noise ratio (SNR) against the original song as shown in Table 2. SNR is a parameter widely used to quantify the level of a signal power to noise, so we use it here to measure the distortion of the adversarial sample over the original song. We then use the following equation $SNR(dB) = 10log_{10}(P_{x(t)}/P_{\delta(t)})$ to obtain SNR, where the original song $x(t)$ is the signal while the perturbation $\delta(t)$ is the noise. Larger SNR value indicates a smaller perturbation. Based on the results in Table 2, the SNR ranges from 14∼18.6 dB, indicating that the perturbation in the original song is less than 4%. Therefore, the perturbation should be too slight to be noticed.

WAA Attack. To practically attack Kaldi over the air, the ideal case is to find a commercial IVC device implemented based on Kaldi and play our adversarial samples against the device. However, we are not aware of any such IVC device, so we simulate a pseudo IVC device based on Kaldi. In particular, the adversarial samples are played by speakers over the air. We use the recording functionality of iPhone 6S to record the audio, which is sent to Kaldi API to decode. Overall, such a pseudo IVC device is built using the microphone in iPhone 6S as the audio recorder, and Kaldi system to decode the audio.

We conducted the practical WAA attack in a meeting room (16 meter long, 8 meter wide, and 4 meter tall). The songs were played using three different speakers including a JBL clip2 portable speaker, an ASUS laptop and a SENMATE broadcast equipment [16], to examine the effectiveness of the injected random noise. All of the speakers are easy to obtain and carry. The distance between the speaker and the pseudo IVC device (i.e., the microphone of the iPhone 6S) was set at 1.5 meters. We chose two commands as in Table 3, and generated adversarial samples. Then we played them over the air using those three different speakers and used the iPhone 6S to record the audios, which were sent to Kaldi to decode. Table 3 shows the WAA attack results. For both of the two commands, JBL speaker overwhelms the other two with the success rate up to 96%, which might indicate its sound quality is better than the other two. All the SNRs are below 2 dB, which indicates slightly bigger perturbation to the original songs due to the random noise from the signal's point of view. Below we will evaluate if such "bigger" perturbation is human-noticeable by conducting a survey.

Human comprehension from the survey. To evaluate the effectiveness of hiding the desired command in the song, we conducted a survey on Amazon Mechanical Turk

Table 3: WAA attack results.

Command	Speaker	Success rate (%)	SNR (dB)	Efficiency (frames/hours)
Echo ask capital one to make a credit card card payment.	JBL speaker	90	1.7	379/2.0
	ASUS Laptop	82	1.7	
	SENMATE Broadcast	72	1.7	
Okay google call one one zero one one nine one two zero.	JBL speaker	96	1.3	400/1.8
	ASUS Laptop	60	1.3	
	SENMATE Broadcast	70	1.3	

(MTurk) [2], an online marketplace for crowdsourcing intelligence. We recruited 204 individuals to participate in our survey[6]. Each participant was asked to listen to 26 adversarial samples, each lasting for about 20 seconds (only about four or five seconds in the middle is crafted to contain the desired command.). A series of questions regarding each audio need to be answered, e.g., (1) whether they have heard the original song before; (2) whether they heard anything abnormal than a regular song (The four options are *no*, *not sure*, *noisy*, and *words different than lyrics*); (3) if choosing *noisy* option in (2), where they believe the noise comes from, while if choosing *words different than lyrics* option in (2), they are asked to write down those words, and how many times they listened to the song before they can recognize the words.

Table 4: Human comprehension of the WTA samples.

Music Classification	Listened (%)	Abnormal (%)	Recognize Command (%)
Soft Music	13	15	0
Rock	33	28	0
Popular	32	26	0
Rap	41	23	0

The entire survey lasts for about five to six minutes. Each participant is compensated \$0.3 for successfully completing the study, provided they pass the attention check question to motivate the participants concentrate on the study. Based on our study, 63.7% of the participants are in the age of 20∼40 and 33.3% are 40∼60 years old, and 70.6% of them use IVC devices (e.g., Amazon Echo, Google home, Smartphone, etc.) everyday.

Table 4 shows the results of the human comprehension of our WTA samples. We show the average results for songs belonging to the same category. The detailed results for each individual song can be referred to in Table 7 in Appendix. Generally, the songs in soft music category are the best candidates for the carrier of the desired command, with as low as 15% of participants noticed the abnormality. None of the participants could recognize any word of the desired command injected in the adversarial samples of any category. Table 5 demonstrates the results of the human comprehension of our WAA samples. On average, 40% of the participants believed the noise was generated by the speaker or like radio, while only 2.2% of them thought the noise from the samples themselves. In addition, less than 1% believed that there were other words except the original lyrics. However, none of them successfully identified any word even repeating the songs several times.

[6]The survey will not cause any potential risks to the participants (physical, psychological, social, legal, etc.). The questions in our survey do not involve any confidential information about the participants. We obtained the IRB Exempt certificates from our institutes.

5.3 Towards the Transferability

Finally, we assess whether the proposed CommanderSong can be transfered to other ASR platforms.

Transfer from Kaldi to iFLYTEK. We choose iFLYTEK ASR system as the target of our transfer, due to its popularity. As one of the top five ASR systems in the world, it possesses 70% of the market in China. Some applications supported by iFLYTEK and their downloads on Google Play as well as the number of worldwide users are listed in Table 8 in Appendix. In particular, *iFLYTEK Input* is a popular mobile voice input method, which supports mandarin, English and personalized input [12]. *iFLYREC* is an online service offered by iFLYTEK to convert audio to text [10]. We use them to test the transferability of our WAA attack samples, and the success rates of different commands are shown in Table 6. Note

Table 5: Human comprehension of the WAA samples.

Song Name	Listened (%)	Abnormal (%)	Noise-speaker (%)	Noise-song (%)
Did You Need It	15	67	42	1
Outlaw of Love	11	63	36	2
The Saltwater Room	27	67	39	3
Sleepwalker	13	67	41	0
Underneath	13	68	45	3
Feeling Good	38	59	36	4
Average	19.5	65.2	40	2.2

Table 6: Transferability from Kaldi to iFLYTEK.

Command	**iFLYREC** (%)	**iFLYTEK Input** (%)
Airplane mode on.	66	0
Open the door.	100	100
Good night.	100	100

that WAA audio samples are directly fed to *iFLYREC* to decode. Meanwhile, they are played using Bose Companion 2 speaker towards *iFLYTEK Input* running on smartphone LG V20, or using JBL speaker towards *iFLYTEK Input* running on smartphone Huawei honor 8/MI note3/iPhone 6S. Those adversarial samples containing commands like *open the door* or *good night* can achieve great transferability on both platforms. However, the command *airplane mode on* only gets 66% success rate on *iFLYREC*, and 0 on *iFLYTEK Input*.

Transferability from Kaldi to DeepSpeech. We also try to transfer CommanderSong from Kaldi to DeepSpeech, which is an open source end-to-end ASR system. We directly fed several adversarial WTA and WAA attack samples to DeepSpeech, but none of them can be decoded correctly. As Carlini et al. have successfully modified any audio into a command recognizable by DeepSpeech [23], we intend to leverage their open source algorithm to examine if it is possible to generate one adversarial sample against both two platforms. In this experiment, we started by 10 adversarial samples generated by CommanderSong, either WTA or WAA attack, integrating commands like *Okay google call one one zero one one nine one two zero*, *Echo open the front door*, and *Echo turn off the light*. We applied their algorithm to modify the samples until DeepSpeech can decode the target commands correctly. Then we tested such newly generated samples against Kaldi as WTA attack, and Kaldi can still successfully recognize them. We did not perform WAA attack since their algorithm targeting DeepSpeech cannot achieve attacks over the air.

The preliminary evaluations on transferability give us the opportunities to understand CommanderSongs and for designing systematic approach to transfer in the future.

5.4 Automated Spreading

Since our WAA attack samples can be used to launch the practical adversarial attack against ASR systems, we want to explore the potential channels that can be leveraged to impact a large amount of victims automatically.

Online sharing. We consider the online sharing platforms like YouTube to spread CommanderSong. We picked up one five-second adversarial sample embedded with the command "*open the door*" and applied Windows Movie Maker software to make a video, since YouTube only supports video uploading. The sample was repeated four times to make the full video around 20 seconds. We then connected our desktop audio output to Bose Companion 2 speaker and installed *iFLYTEK Input* on LG V20 smartphone. In this experiment, the distance between the speaker and the phone can be up to 0.5 meter, and *iFLYTEK Input* can still decode the command successfully.

Radio broadcasting. In this experiment, we used HackRF One [8], a hardware that supports Software Defined Radio (SDR) to broadcast our CommanderSong at the frequency of FM 103.4 MHz, simulating a radio station. We setup a radio at the corresponding frequency, so it can receive and play the CommanderSong. We ran the *WeChat*[7] application and enabled the *iFLYTEK Input* on different smartphones including iPhone 6S, Huawei Honor 8 and XiaoMi MI Note3. *iFLYTEK Input* can always successfully recognize the command "*open the door*" from the audio played by the radio and display it on the screen.

5.5 Efficiency

We also evaluate the cost of generating CommanderSong in the aspect of the required time. For each command, we record the time to inject it into different songs and compute the average. Since the time required to craft also depends on the length of the desired command, we define the efficiency as the ratio of the number of frames of the desired command and the required time. Table 2 and Table 3 show the efficiency of generating WTA and WAA samples for different commands. Most of those adversarial samples can be generated in less than two hours, and some simple commands like "*Echo open the front door*" can be done within half an hour. However, we do notice that some special words (such as *GPS* and *airplane*) in the command make the generation time longer. Probably those words are not commonly used in the training process of the "ASpIRE model" of Kaldi, so generating enough phonemes to represent the words is time-consuming. Furthermore, we find that, for some songs in the rock category such as "*Bang bang*" and "*Roaked*", it usually takes longer to generate the adversarial samples for the same command compared with the songs in other categories, probably due to the unstable rhythm of them.

6 Understanding the Attacks

We try to deeply understand the attacks, which could potentially help to derive defense approaches. We raise some

[7]*WeChat* is the most popular instant messaging application in China, with approximately 963,000,000 users all over the world by June 2017 [15].

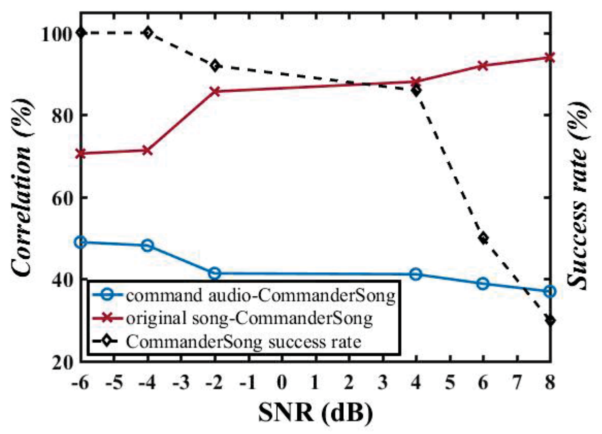


Figure 4: SNR impacts on correlation of the audios and the success rate of adversarial audios.

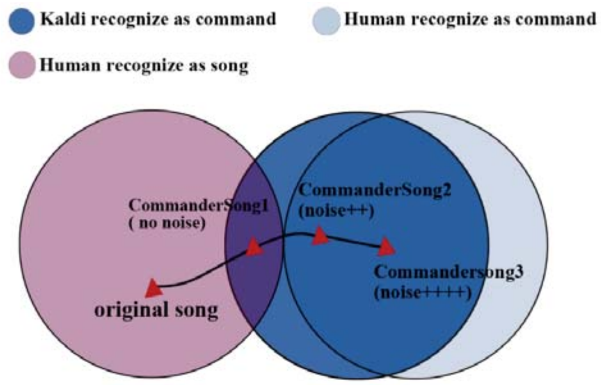

Kaldi recognize as command
Human recognize as command
Human recognize as song
CommanderSong1 (no noise)
CommanderSong2 (noise++)
Commandersong3 (noise++++)
original song

Figure 5: Explaination of Kaldi and human recognition of the audios.

questions and perform further analysis on the attacks.

In what ways does the song help the attack? We use songs as the carrier of commands to attack ASR systems. Obviously, one benefit of using a song is to prevent listeners from being aware of the attack. Also CommanderSong can be easily spread through Youtube, radio, TV, etc. Does the song itself help generate the adversarial audio samples? To answer this question, we use a piece of silent audio as the "carrier" to generate CommanderSong A_{cs} (WAA attack), and test the effectiveness of it. The results show that A_{cs} can work, which is aligned with our findings – a random song can serve as the "carrier" because a piece of silent audio can be viewed as a special song.

However, after listening to A_{cs}, we find that A_{cs} sounds quite similar to the injected command, which means any user can easily notice it, so A_{cs} is not the adversarial samples we desire. Note that, in our human subject study, none of the participants recognized any command from the generated CommanderSongs. We assume that *some phonemes or even smaller units in the original song work together with the injected small perturbations to form the target command*. To verify this assumption, we prepare a song A_s and use it to generate the CommanderSong A_{cs}. Then we calculate the difference $\Delta(A_s, A_{cs})$ between them, and try to attack ASR systems using $\Delta(A_s, A_{cs})$. However, after several times of testing, we find that $\Delta(A_s, A_{cs})$ does not work, which indicates the pure perturbations we injected cannot be recognized as the target commands.

Recall that in Table 5, the songs in the soft music category are proven to be the best carrier, with lowest abnormality identified by participants. Based on the findings above, it appears that such songs can better aligned with the phonemes or smaller "units" in the target command to help the attack. This is also the reason why $\Delta(A_s, A_{cs})$ cannot directly attack successfully: the "units" in the song combined with $\Delta(A_s, A_{cs})$ together construct the phonemes of the target command.

What is the impact of noise in generating adversarial samples? As mentioned early, we build a generic random noise model to perform the WAA attack over the air. In order to understand the impact of the noise in generating adversarial samples, we crafted CommanderSong using noises with different amplitude values. Then we observed the differences between the CommanderSong and the original song, the differences between the CommanderSong and the pure command audio, and the success rates of the CommanderSong to attack. To characterize the difference, we leverage Spearman's rank correlation coefficient [46] (*Spearman's rho* for short) to represent the similarity between two pieces of audio. Spearman's rho is widely used to represent the correlation between two variables, and can be calculated as follows: $r(X,Y) = Cov(X,Y)/\sqrt{Var[X]Var[Y]}$, where X and Y are the MFCC features of the two pieces of audio. $Cov(X,Y)$ represents the covariance of X and Y. $Var[X]$ and $Var[Y]$ are the variances of X and Y respectively.

The results are shown in Figure 4. The x-axis in the figure shows the *SNR* (in *dB*) of the noise, and the y-axis gives the correlation. From the figure, we find that the correlation between the CommanderSong and the original song (red line) decreases with *SNR*. It means that the CommanderSong sounds less like the original song when the amplitude value of the noise becomes larger. This is mainly because the original song has to be modified more to find a CommanderSong robust enough against the introduced noise. On the contrary, the CommanderSong becomes more similar with the target command audio when the amplitude values of the noise increases (i.e., decrease of *SNR* in the figure, blue line), which means that the CommanderSong sounds more like the target command. The success rate (black dotted line) also increases with the decrease of SNR. We also note that, when

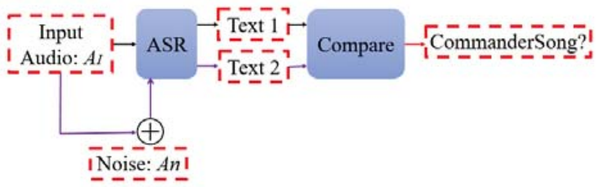


Figure 6: Audio turbulence defense.

$SNR = 4\ dB$, the success rate could be as high as 88%. Also the correlation between CommanderSong and the original song is 90%, which indicates a high similarity.

Figure 5 shows the results from another perspective. Suppose the dark blue circle is the set of audios that can be recognized as commands by ASR systems, while the light blue circle and the red one represent the sets of audio recognized as commands and songs by human respectively. At first, the original song is in the red circle, which means that neither ASR systems nor human being recognize any command inside. WTA attack slightly modifies the song so that the open source system Kaldi can recognize the command while human cannot. After noises are introduced to generate CommanderSong for WAA attacks, CommanderSong will fall into the light blue area step by step, and in the end be recognized by human. Therefore, attackers can choose the amplitude values of noise to balance between robustness to noise and identifiability by human users.

7 Defense

We propose two approaches to defend against CommanderSong: Audio turbulence and Audio squeezing. The first defense is effective against WTA, but not WAA; while the second defense works against both attacks.

Audio turbulence. From the evaluation, we observe that noise (e.g., from speaker or background) decreases the success rate of CommanderSong while impacts little on the recognition of audio command. So our basic idea is to add noise (referred to as *turbulence* noise A_n) to the input audio A_I before it is received by the ASR system, and check whether the resultant audio $A_I \bigoplus A_n$ can be interpreted as other words. Particularly, as shown in Figure 6, A_I is decoded as $\texttt{text}_1$ by the ASR system. Then we add A_n to A_I and let the ASR system extract the text $\texttt{text}_2$ from $A_I \bigoplus A_n$. If $\texttt{text}_1 \neq \texttt{text}_2$, we say that the CommanderSong is detected.

We did experiments using this approach to test the effectiveness of such defense. The target command "open the door" was used to generate a CommanderSong. Figure 7 shows the result. The x-axis shows the *SNR* (A_I to A_n), and the y-axis shows the success rate. We found that the success rate of WTA dramatically drops when *SNR*

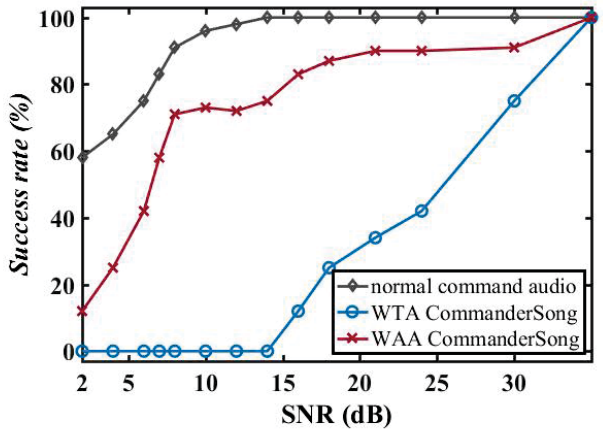


Figure 7: The results of audio turbulence defense.

decreases. When $SNR = 15\ dB$, WTA almost always fails and A_I can still be successfully recognized, which means this approach works for WTA. However, the success rate of WAA is still very high. This is mainly because CommanderSongs for WAA is generated using random noises, which is robust against turbulence noise.

Audio squeezing. The second defense is to reduce the sampling rate of the input audio A_I (just like squeezing the audio). Instead of adding A_n in the defense of audio turbulence, we downsample A_I (referred to as $D(A_I)$). Still, ASR systems decode A_I and $D(A_I)$, and get $\texttt{text}_1$ and $\texttt{text}_2$ respectively. If $\texttt{text}_1 \neq \texttt{text}_2$, the CommanderSong is detected. Similar to the previous experiment, we evaluate the effectiveness of this approach. The results are shown in Figure 8. The x-axis shows the ratio ($1/M$) of downsampling (M is the downsampling factor or decimation factor, which means that the original sampling rate is M times of the downsampled rate). When $1/M = 0.7$ (if the sample rate is 8000 samples/second, the downsampled rate is 5600 samples/second), the success rates of WTA and WAA are 0% and 8% respectively. A_I can still be successful recognized at the rate of 91%. This means that Audio squeezing is effective to defend against both WTA and WAA.

8 Related Work

Attack on ASR system. Prior to our work, many researchers have devoted to security issues about speech controllable systems [36, 35, 26, 41, 51, 22, 53, 23]. Denis et al. found the vulnerability of analog sensor and injected bogus voice signal to attack the microphone [36]. Kasmi et al. stated that, by leveraging intentional electromagnetic interference on headset cables, voice command could be injected and carried by FM signals which is further received and interpreted by smart phones [35].

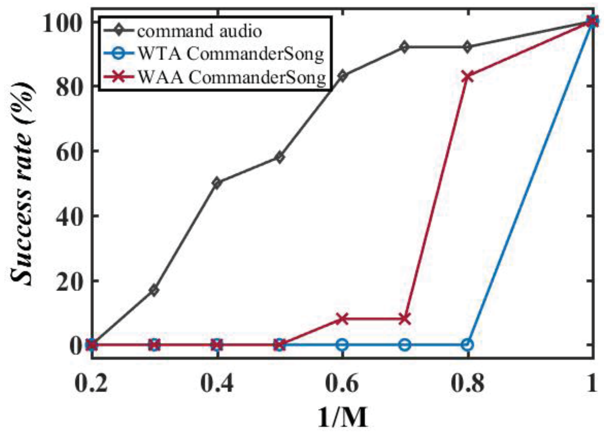


Figure 8: Audio squeezing defense result.

Diao et al. demonstrated that, through permission bypassing attack in Android smart phones, voice commands could be played using apps with zero permissions [26]. Mukhopadhyay et al. considered voice impersonation attacks to contaminate a voice-based user authentication system [41]. They reconstructed the victims voice model from the victims voice data, and launched attacks that can bypass voice authentication systems. Different from these attacks, we are attacking the machine learning models of ASR systems.

Hidden voice command [22] launched both black box (i.e., inverse MFCC) and white box (i.e., gradient decent) attacks against ASR systems with GMM-based acoustic models. Different from this work, our target is a DNN-based ASR system. Recently, the authors posted the achievement that can construct targeted audio adversarial examples on DeepSpeech, an end-to-end open source ASR platform [23]. To perform the attack, the adversary needs to directly upload the adversarial WAV file to the speech recognition system. Our attacks on Kaldi are concurrent to their work, and our attack approaches are independent to theirs. Moreover, our attacks succeed under a more practical setting that let the adversarial audio be played over the air. The recent work DolphinAttack [53] proposed a completely inaudible voice attack by modulating commands on ultrasound carriers and leveraging microphone vulnerabilities to attack. As noted by the authors, such attack can be eliminated by filtering out ultrasound carrier (e.g., iPhone 6 Plus). Differently, our attack uses songs instead of ultrasound as the carriers, making the attack harder to defend.

Adversarial research on machine learning. Besides attacking speech recognition systems, there has been substantial work on adversarial machine learning examples towards physical world. Kurakin et al. [37] proved it is doable that Inception v3 image classification neural network could be compromised by adversarial images. Brown et al. [21] showed by adding an universal patch to an image they could fool the image classifiers successfully. Evtimov et al. [27] proposed a general algorithm which can produce robust adversarial perturbations into images to overcome physical condition in real world. They successfully fooled road sign classifiers to mis-classify real Stop Sign. Different from them, our study targets speech recognition system.

Defense of Adversarial on machine learning. Defending against adversarial attacks is known to be a challenging problem. Existing defenses include adversarial training and defensive distillation. Adversarial training [39] adds the adversarial examples into the model's training set to increase its robustness against these examples. Defensive distillation [33] trains the model with probabilities of different class labels supported by an early model trained on the same task. Both defenses perform a kind of gradient masking [45] which increases the difficulties for the adversary to compute the gradient direction. In [29], Dawn Song attempted to combine multiple defenses including feature squeezing and the specialist to construct a larger strong defense. They stated that defenses should be evaluated by strong attacks and adaptive adversarial examples. Most of these defenses are effective for white box attacks but not for black box ones. Binary classification is another simple and effective defense for white box attacks without any modifications of the underlying systems. A binary classifier is built to separate adversarial examples apart from the clean data. Similar as adversarial training and defensive distillation, this defense suffers from generalization limitation. In this paper, we propose two novel defenses against CommanderSong attack.

9 Conclusion

In this paper, we perform practical adversarial attacks on ASR systems by injecting "voice" commands into songs (CommanderSong). To the best of our knowledge, this is the first systematical approach to generate such practical attacks against DNN-based ASR system. Such CommanderSong could let ASR systems execute the command while being played over the air without notice by users. Our evaluation shows that CommanderSong can be transferred to iFLYTEK, impacting popular apps such as WeChat, Sina Weibo, and JD with billions of users. We also demonstrated that CommanderSong can be spread through YouTube and radio. Two approaches (audio turbulence and audio squeezing) are proposed to defend against CommanderSong.

Acknowledgments

IIE authors are supported in part by National Key R&D Program of China (No.2016QY04W0805), NSFC U1536106, 61728209, National Top-notch Youth Talents Program of China, Youth Innovation Promotion Association CAS and Beijing Nova Program. Indiana University author is supported in part by the NSF 1408874, 1527141, 1618493 and ARO W911NF1610127. Illinois University authors are supported in part by NSF CNS grants 13-30491, 14-08944, and 15-13939.

References

[1] *Amazon Alexa*. https://developer.amazon.com/alexa.

[2] *Amazon Mechanical Turk*. https://www.mturk.com.

[3] *Apple Siri*. https://www.apple.com/ios/siri.

[4] *Aspire*. https://github.com/kaldi-asr/kaldi/tree/master/egs/aspire.

[5] *CMUSphinx*. https://cmusphinx.github.io/.

[6] *Google Assistant*. https://assistant.google.com.

[7] *Google Text-to-speech*. https://play.google.com/store/apps.

[8] *HackRF One*. https://greatscottgadgets.com/hackrf/.

[9] *HTK*. http://htk.eng.cam.ac.uk/.

[10] *iFLYREC*. https://www.iflyrec.com/.

[11] *iFLYTEK*. http://www.iflytek.com/en/index.html.

[12] *iFLYTEK Input*. http://www.iflytek.com/en/mobile/iflyime.html.

[13] *Kaldi*. http://kaldi-asr.org.

[14] *Microsoft Cortana*. https://www.microsoft.com/en-us/cortana.

[15] *Number of monthly active WeChat users from 2nd quarter 2010 to 2nd quarter 2017 (in millions)*. https://www.statista.com/statistics/255778/number-of-active-wechat-messenger-accounts/.

[16] *SENMATE broadcast*. http://www.114pifa.com/p106/34376.html.

[17] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.

[18] Kartik Audhkhasi, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny. Building competitive direct acoustics-to-word models for english conversational speech recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[19] Wei Bao, Hong Li, Nan Li, and Wei Jiang. A liveness detection method for face recognition based on optical flow field. In *Image Analysis and Signal Processing, 2009. IASP 2009. International Conference on*, pages 233–236. IEEE, 2009.

[20] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[21] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[22] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. Hidden voice commands. In *USENIX Security Symposium*, pages 513–530, 2016.

[23] Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *Deep Learning and Security Workshop*, 2018.

[24] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

[25] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2004.

[26] Wenrui Diao, Xiangyu Liu, Zhe Zhou, and Kehuan Zhang. Your voice assistant is mine: How to abuse speakers to steal information and control your phone. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, pages 63–74. ACM, 2014.

[27] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on deep learning models. *Computer Vision and Pattern Recognition*, 2018.

[28] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

[29] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defenses: Ensembles of weak defenses are not strong. *USENIX Workshop on Offensive Technologies*, 2017.

[30] Hynek Hermansky. Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[31] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[32] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[33] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*, 2014.

[34] Fumitada Itakura. Line spectrum representation of linear predictor coefficients of speech signals. *The Journal of the Acoustical Society of America*, 57(S1):S35–S35, 1975.

[35] Chaouki Kasmi and Jose Lopes Esteves. Iemi threats for information security: Remote command injection on modern smartphones. *IEEE Transactions on Electromagnetic Compatibility*, 57(6):1752–1755, 2015.

[36] Denis Foo Kune, John Backes, Shane S Clark, Daniel Kramer, Matthew Reynolds, Kevin Fu, Yongdae Kim, and Wenyuan Xu. Ghost talk: Mitigating emi signal injection attacks against analog sensors. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 145–159. IEEE, 2013.

[37] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[38] Pan Li, Qiang Liu, Wentao Zhao, Dongxu Wang, and Siqi Wang. BEBP: an poisoning method against machine learning based idss. *arXiv preprint arXiv:1803.03965*, 2018.

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

[40] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. Voice recognition algorithms using mel frequency cepstral coefficient (mfcc) and dynamic time warping (dtw) techniques. *Journal of Computing, Volume 2, Issue 3*, 2010.

[41] Dibya Mukhopadhyay, Maliheh Shirvanian, and Nitesh Saxena. All your voices are belong to us: Stealing voices to fool humans and machines. In *European Symposium on Research in Computer Security*, pages 599–621. Springer, 2015.

[42] Douglas OShaughnessy. Automatic speech recognition: History, methods and challenges. *Pattern Recognition*, 41(10):2965–2979, 2008.

[43] Nicolas Papernot, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, Ryan Sheatsley, et al. cleverhans v2. 0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

[44] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[45] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[46] W Pirie. Spearman rank correlation coefficient. *Encyclopedia of statistical sciences*, 1988.

[47] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar. Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer. In *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*, pages 193–199. IEEE, 2017.

[48] Stephanie AC Schuckers. Spoofing and anti-spoofing measures. *Information Security technical report*, 7(4):56–62, 2002.

[49] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[50] Roberto Tronci, Daniele Muntoni, Gianluca Fadda, Maurizio Pili, Nicola Sirena, Gabriele Murgia, Marco Ristori, Sardegna Ricerche, and Fabio Roli. Fusion of multiple clues for photo-attack detection in face recognition systems. In *Biometrics (IJCB), 2011 International Joint Conference on*, pages 1–6. IEEE, 2011.

[51] Tavish Vaidya, Yuankai Zhang, Micah Sherr, and Clay Shields. Cocaine noodles: exploiting the gap between human and machine speech recognition. *WOOT*, 15:10–11, 2015.

[52] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. The microsoft 2016 conversational speech recognition system. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 5255–5259. IEEE, 2017.

[53] Guoming Zhang, Chen Yan, Xiaoyu Ji, Tianchen Zhang, Taimin Zhang, and Wenyuan Xu. Dolphinattack: Inaudible voice commands. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 103–117. ACM, 2017.

Appendix

Table 7: The detailed results of individual song in human comprehension survey for WTA samples. When we were checking the survey results from MTurk, we found the average familiarity of MTurk workers towards our songs is not as good as we expected. So streaming counts from Spotify are also listed in the table, as we want to show the popularity of our sample songs. The song *Selling Brick in Street* is not in Spotify database so we can not provide the count for it.

Music Classification	**Song Name**	**Spotify Streaming Count**	**Listened (%)**	**Abnormal (%)**	**Recognize Command (%)**
Soft Music	Heart and Soul	13,749,471	15%	8%	0
	Castle in the Sky	2,332,348	9%	6%	0
	A Comme Amour	1,878,899	14%	18%	0
	Mariage D'amour	337,486	17%	33%	0
	Lotus	49,443,256	11%	12%	0
	Average	13,548,292	13%	15%	0
Rock	Bang Bang	532,057,658	52%	24%	0
	Soaked	29,734	13%	32%	0
	Gold	11,614,629	14%	41%	0
	We are never Getting back together	113,806,946	66%	38%	0
	When can I See You again	26,463,993	20%	9%	0
	Average	136,794,562	33%	28%	0
Popular	Love Story	109,952,344	49%	24%	0
	Hello Seattle	9,850,328	29%	16%	0
	Good Time	125,125,693	48%	32%	0
	To the Sky	4,860,627	27%	30%	0
	A Loaded Smile	658,814	8%	26%	0
	Average	50,089,561	32%	26%	0
Rap	Rap God	349,754,768	43%	32%	0
	Let Me Hold You	311,569,726	31%	15%	0
	Lose Yourself	483,937,007	75%	14%	0
	Remember the Name	193,564,886	48%	32%	0
	Selling Brick in Street	N/A	6%	24%	0
	Average	334,706,597	41%	23%	0

Table 8: The detailed information of some sample applications which utilize iFLYTEK as voice input, including number of downloads from Google Play and total user amount. Since Google Services are not accessible in China and information of Apple App Store is not collected, the number of users may not be associated with the number of downloads in Google Play. As shown in the table, each of these applications has over 0.2 billion users in the world.

Application	**Usage**	**Downloads from Google Play**	**Total Users Worldwide (Billion)**
Sina Weibo	Social platform	11,000,000	0.53
JD	Online shopping	1,000,000	0.27
CMbrowser	Searching engine	50,000,000	0.64
Ctrip	Travel advice website	1,000,000	0.30
Migu Digital	Voice assistant	5,000	0.46
WeChat	Chatting, Social	100,000,000	0.96
iFLYTEK Input	Typing, Voice Input	500,000	0.5